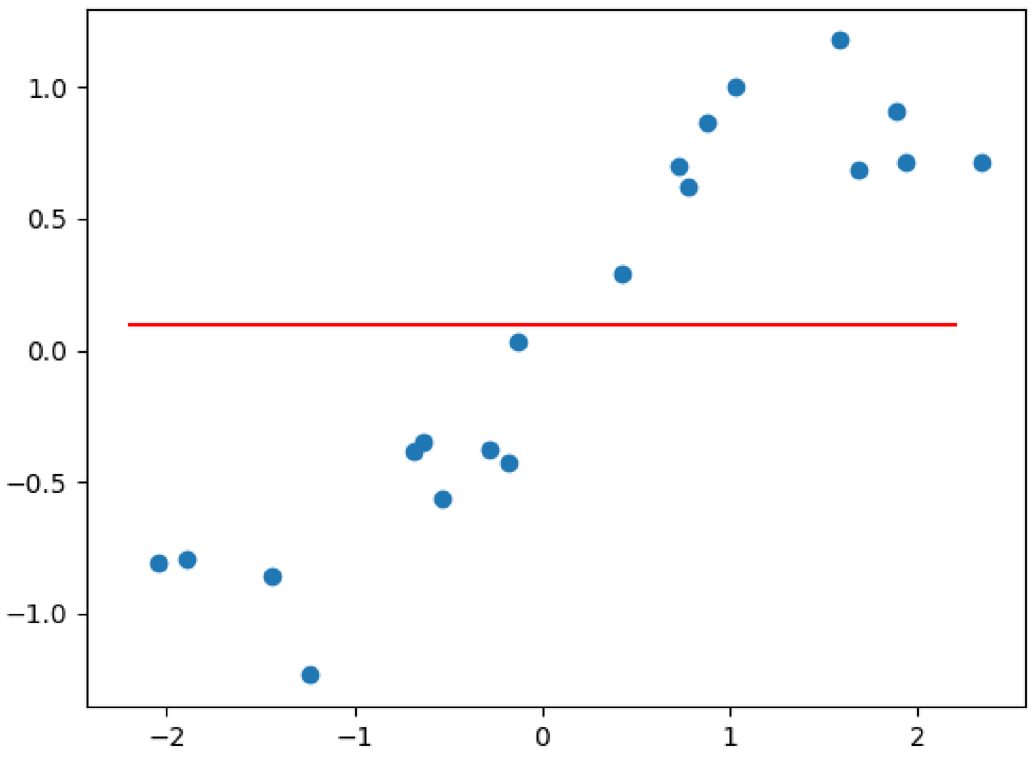
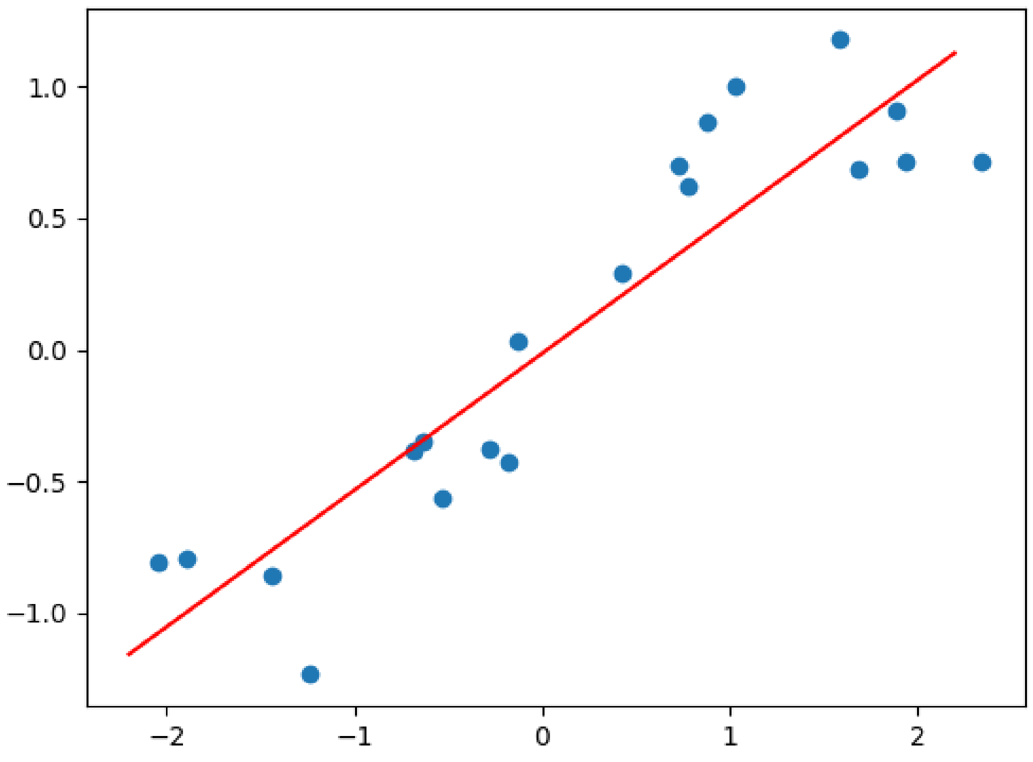
Report:

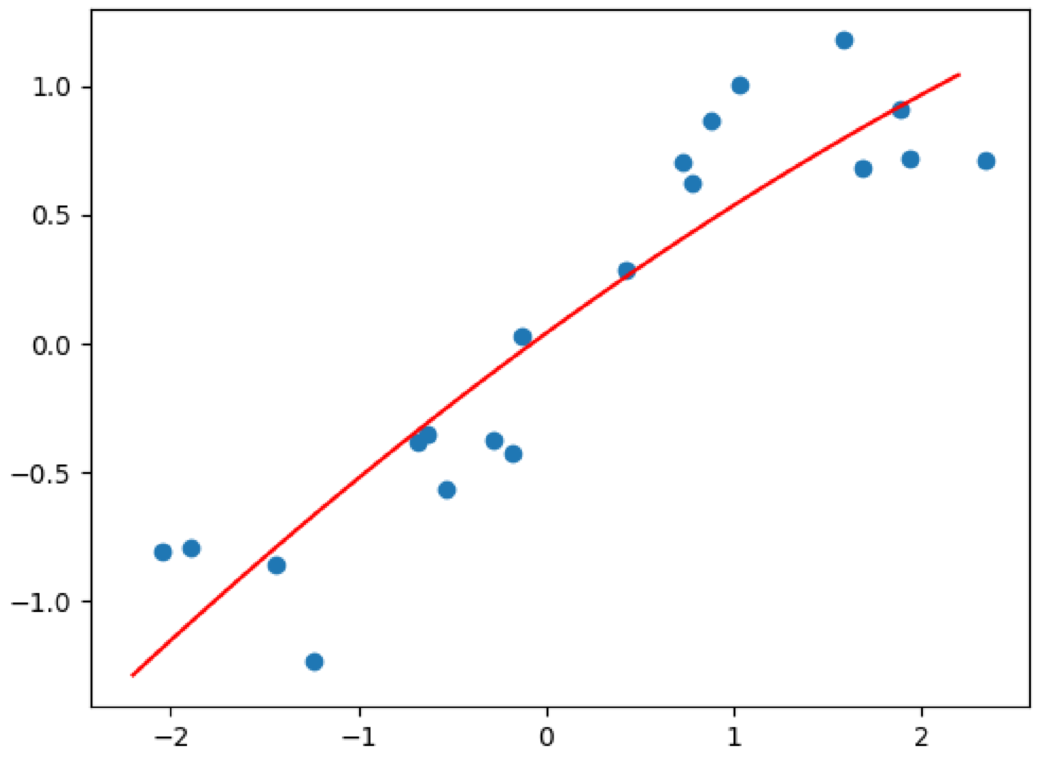
After I plotted the curves with degrees 0 to 6, using only the training data, I have calculated the sum of squared errors for each degree and I have plotted how the error changes with respect to the degree of the polynomial. I observed that the sum of squared errors was decreasing as the degree of the polynomial increased. This was expected, because polynomials with higher degree have more flexibility compared to the ones with lower degrees. That's why they can fit the curve better to the data thus decreased the error.

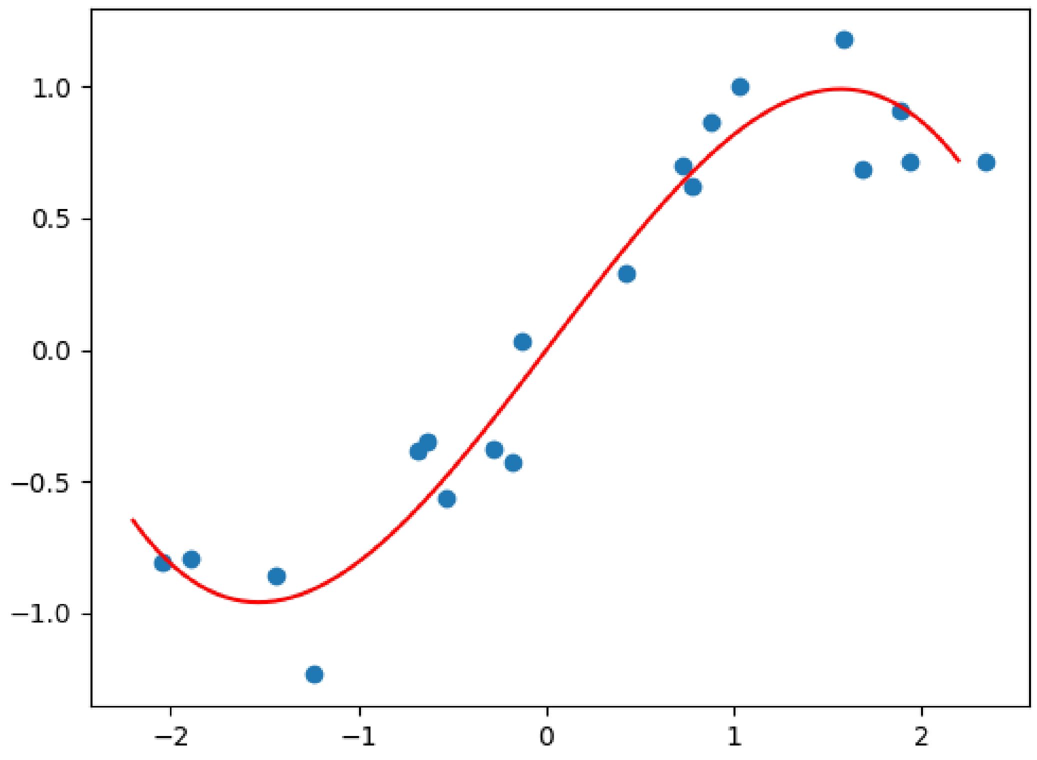
This sounds good, but actually it may lead to overfitting the data. In fact I have observed this problem after calculating the sum of squared errors for each degree using the testing data. Same as before, I have plotted how the error changes with respect to the degree of the polynomial. Sum of squared errors decreased as degree increased in smaller degrees. But as the degree increased the decrease of errors have disappeared and in the end the error started to increase. This clearly shows that by using a polynomial with higher degree, we have overfitted our training set.

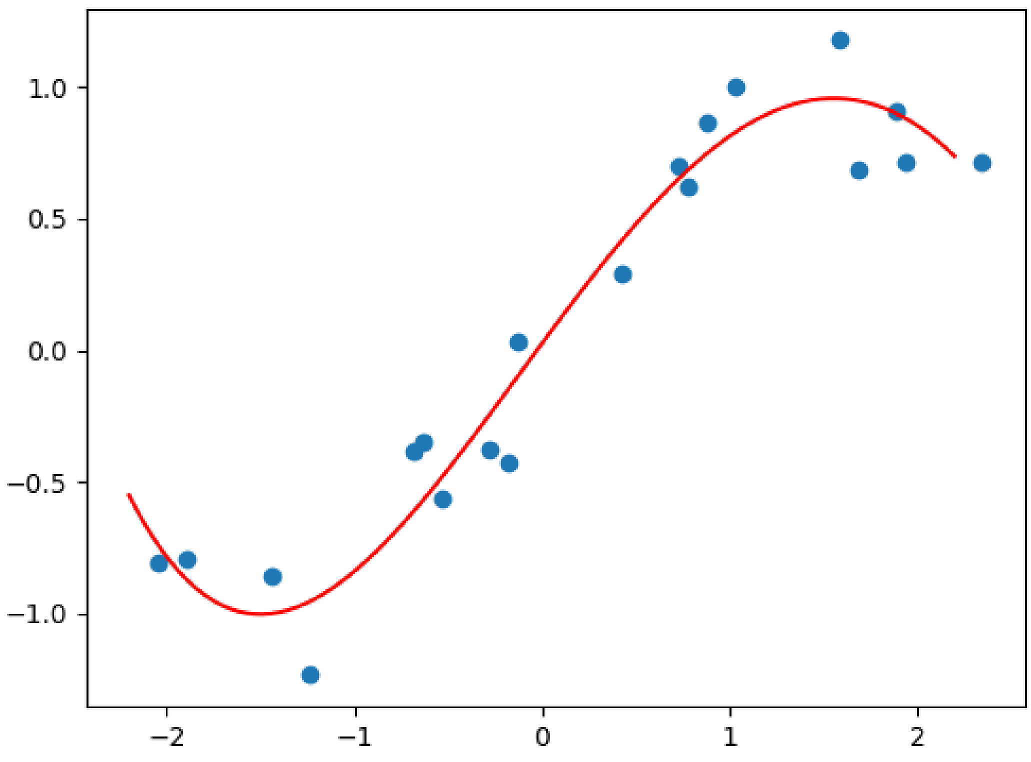
You can see the plots that I have created. The plots are ordered with respect to the degree. First 8 plots are the plots I have created using the training set and the other 8 are the ones I have created using the testing set.


1.0
0.5
0.0
−0.5
−1.0
−2
−1
0
1
2

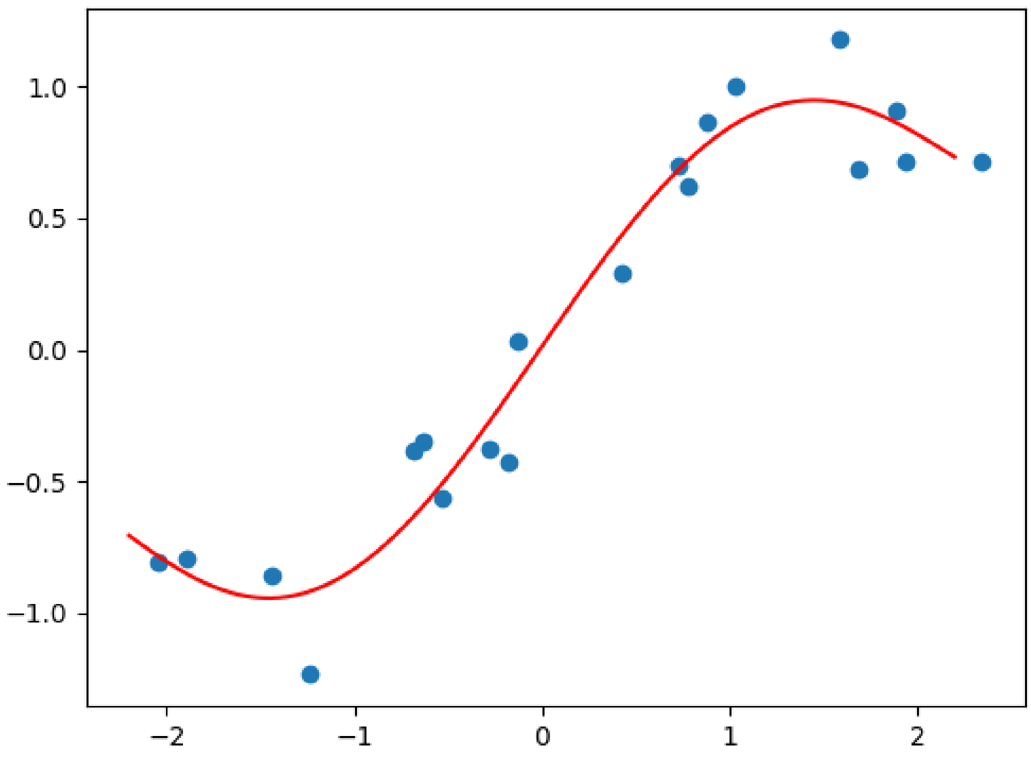

1.0
0.5
0.0
−0.5
−1.0
−2
−1
0
1
2


1.0
0.5
0.0
−0.5
−1.0
−2
−1
0
1
2

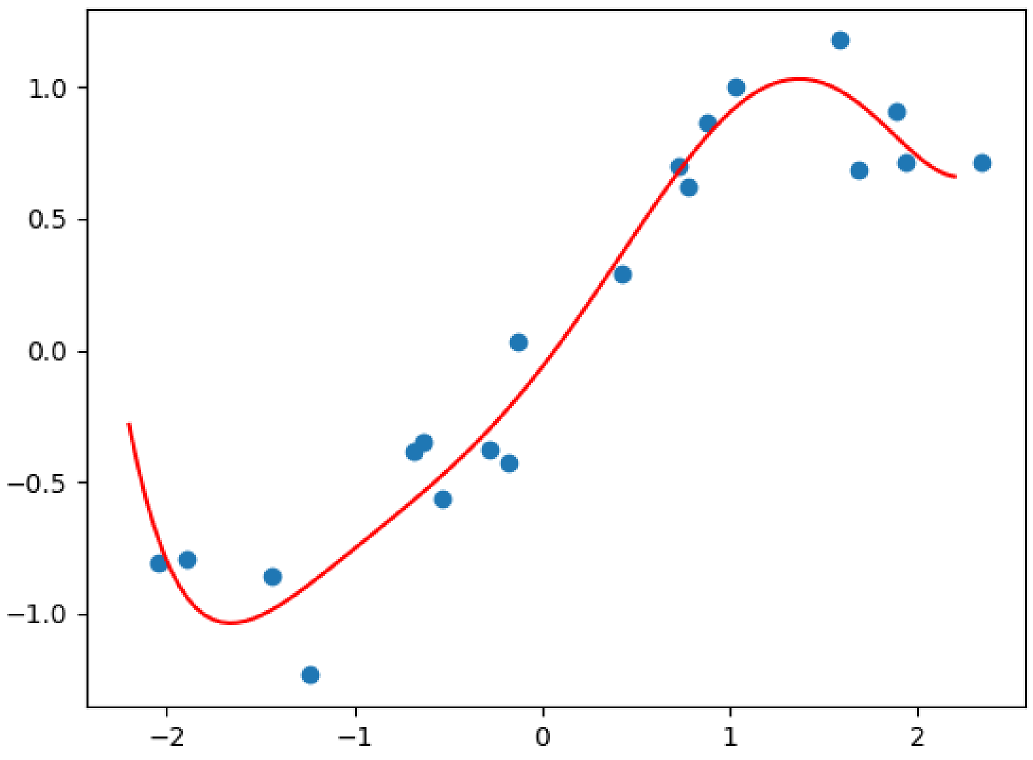

1.0
0.5
0.0
−0.5
−1.0
−2
−1
0
1
2



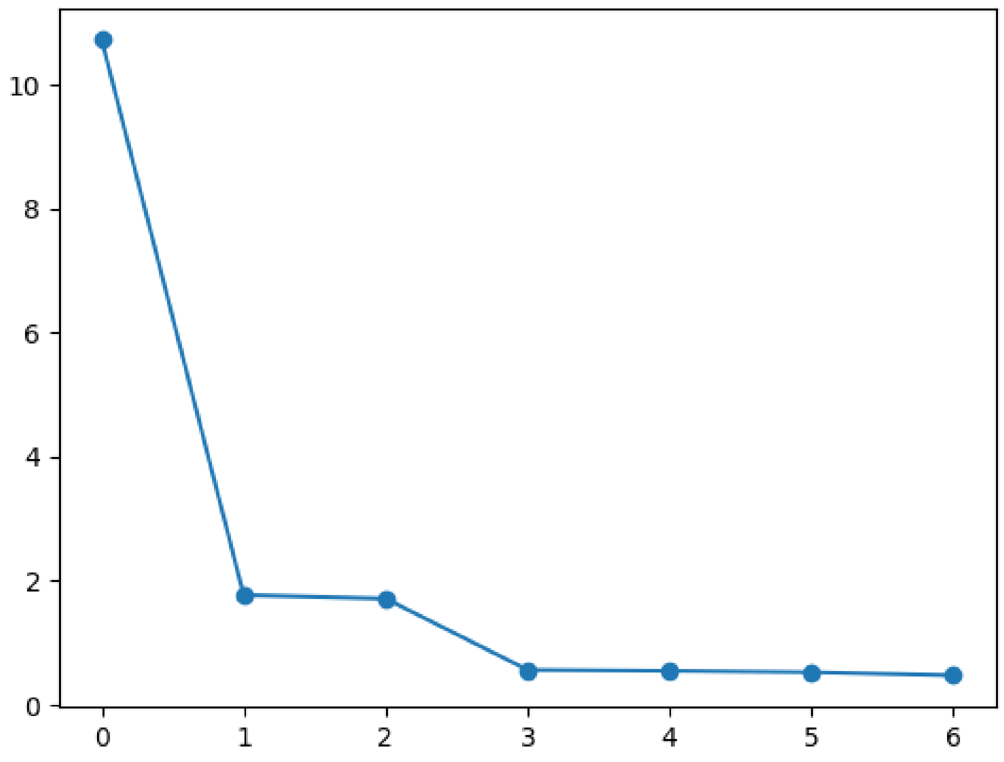
1.0
0.5
0.0
−0.5
−1.0
−2
−1
0
1
2



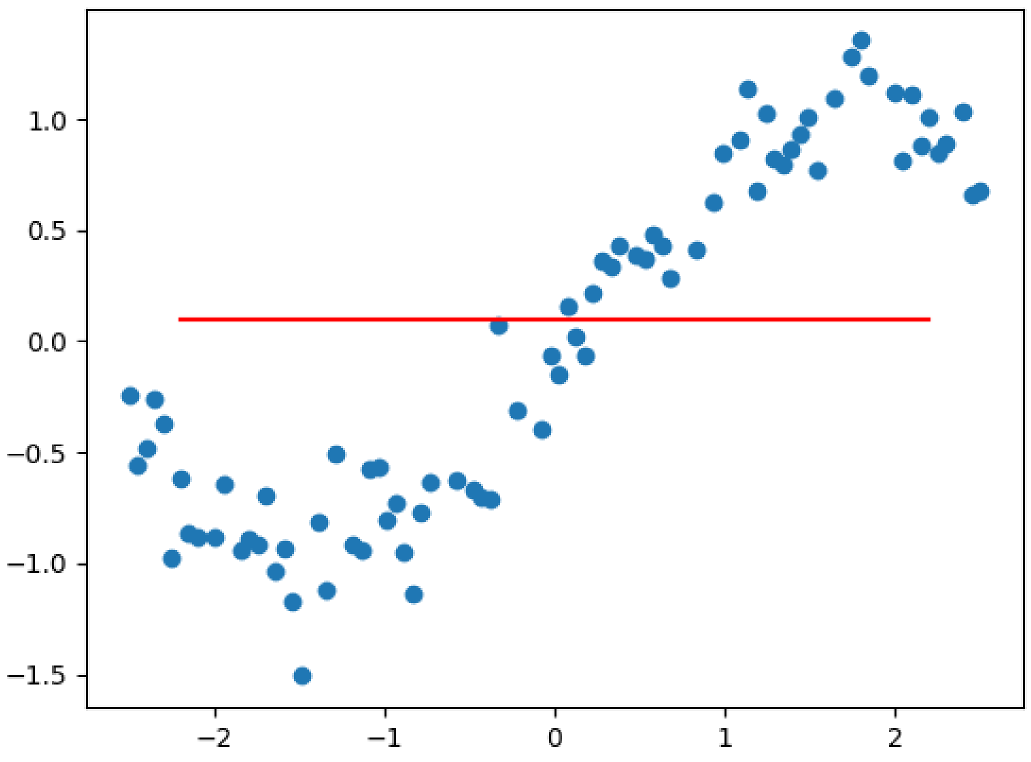
1.0
0.5
0.0
−0.5
−1.0
−2
−1
0
1
2



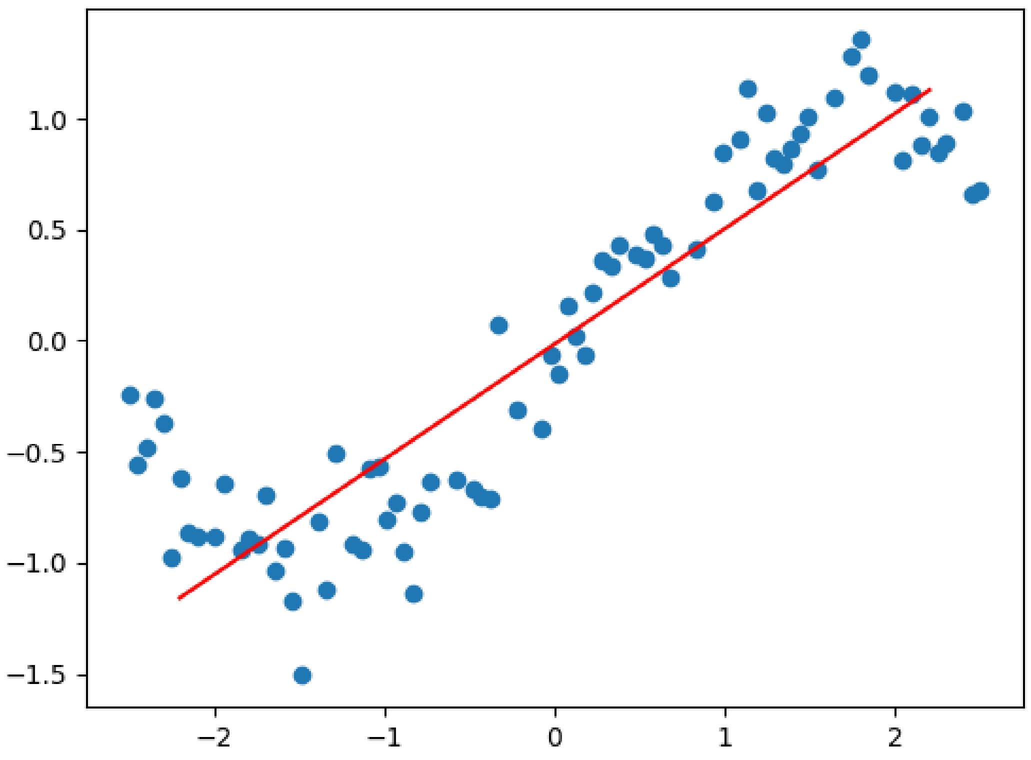
1.0
0.5
0.0
−0.5
−1.0
−2
−1
0
1
2



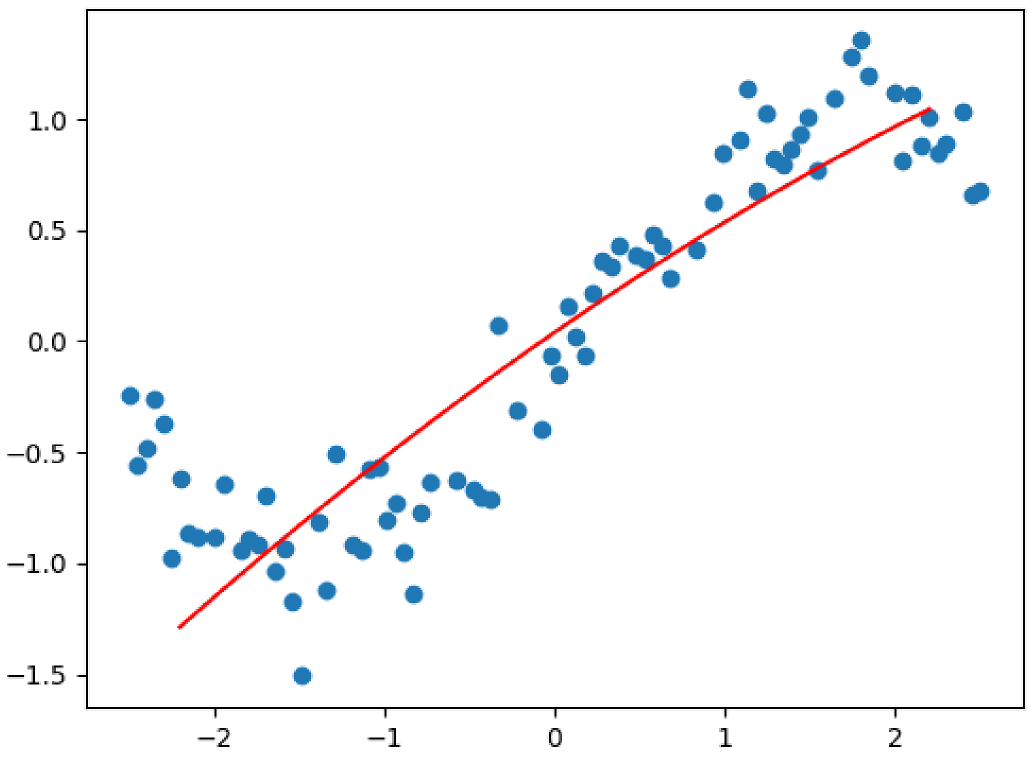
10
8
6
4
2
0
0
1
2
3
4
5
6



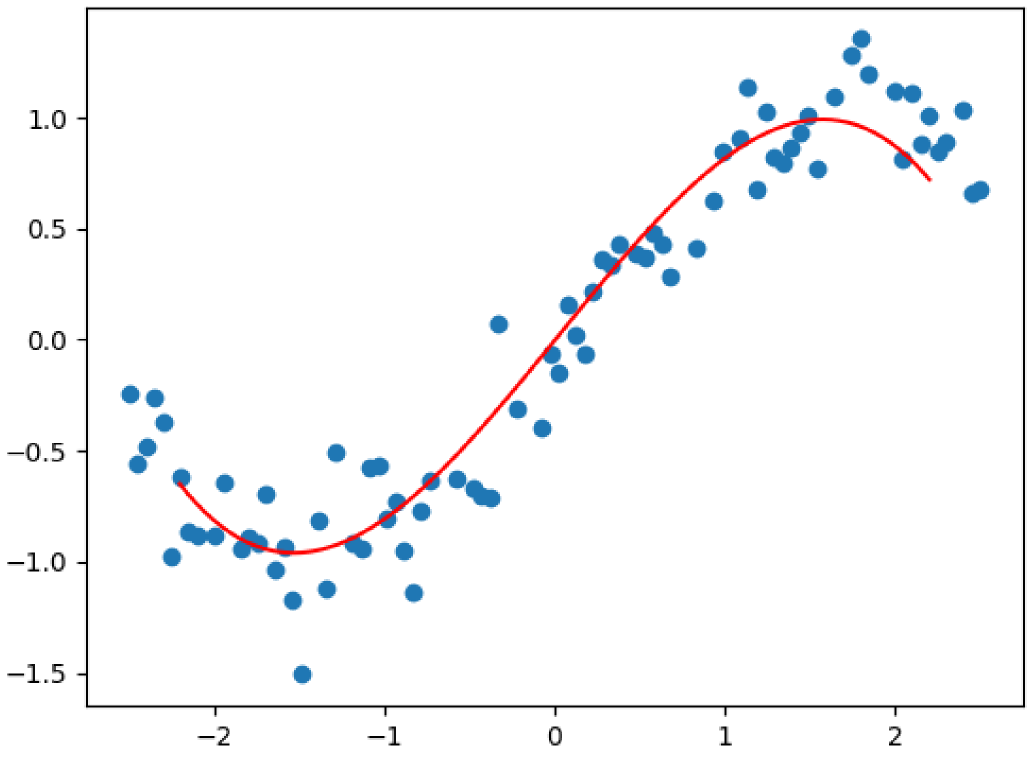
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2
−1
0
1
2



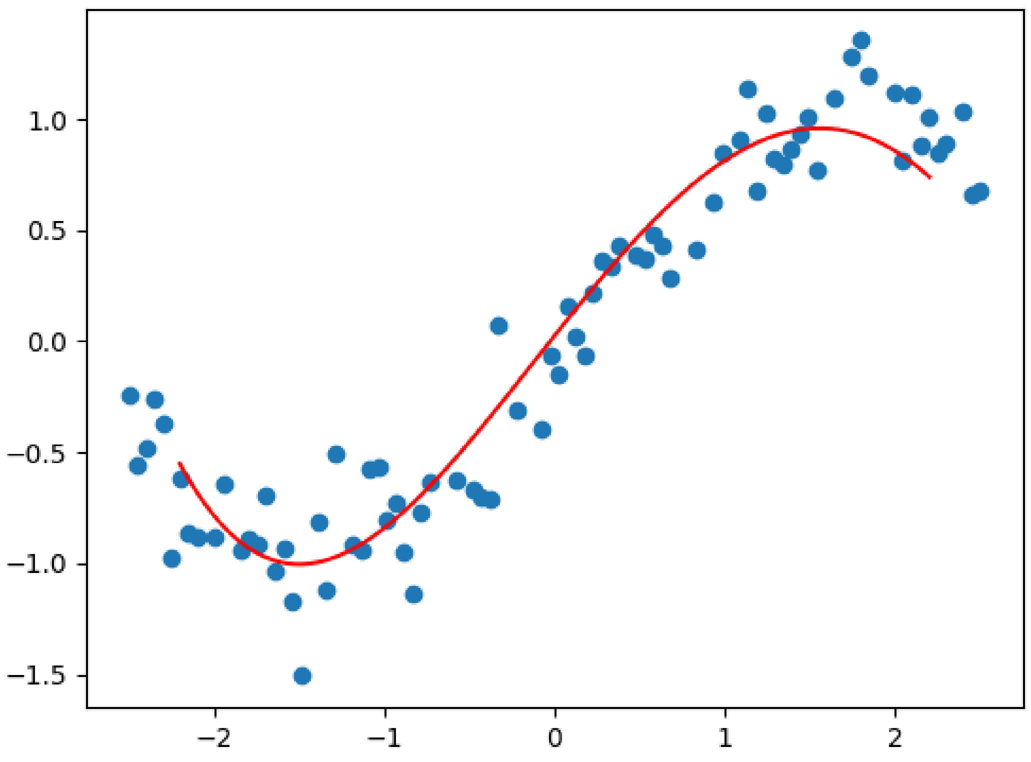
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2
−1
0
1
2



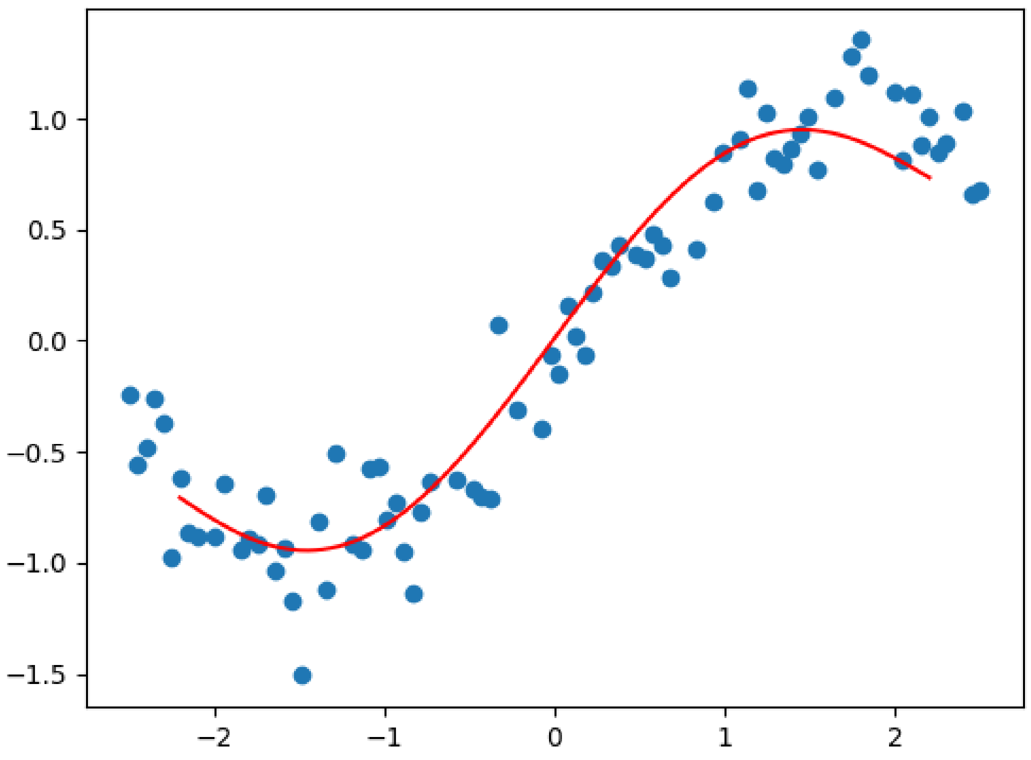
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2
−1
0
1
2



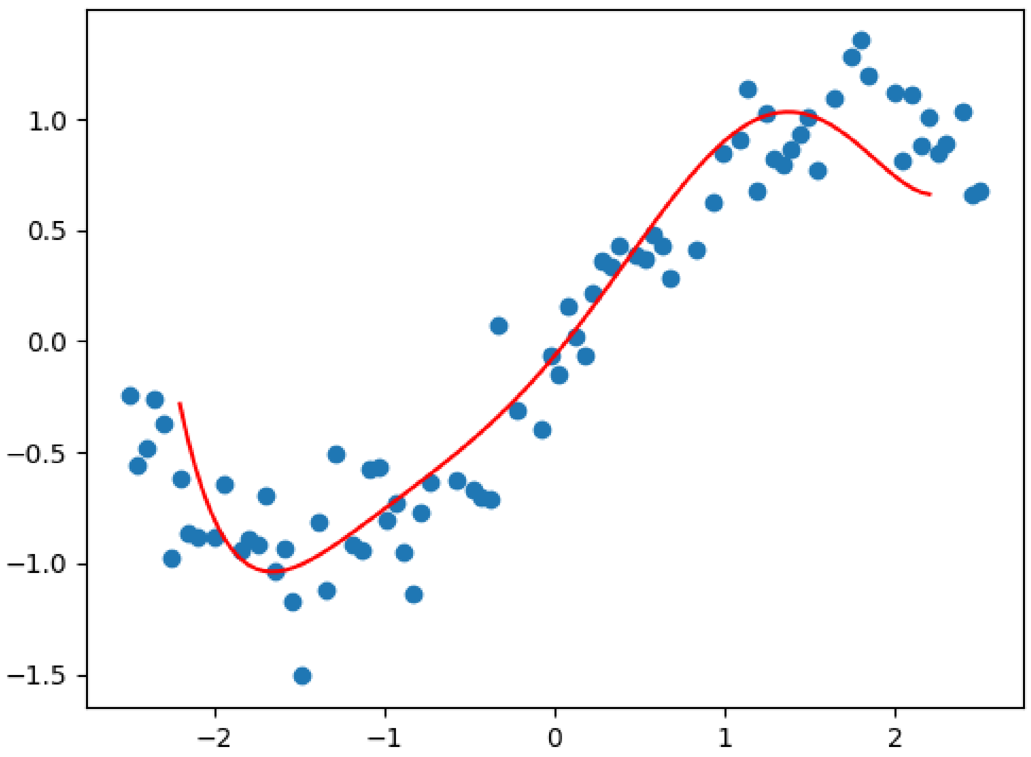
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2
−1
0
1
2



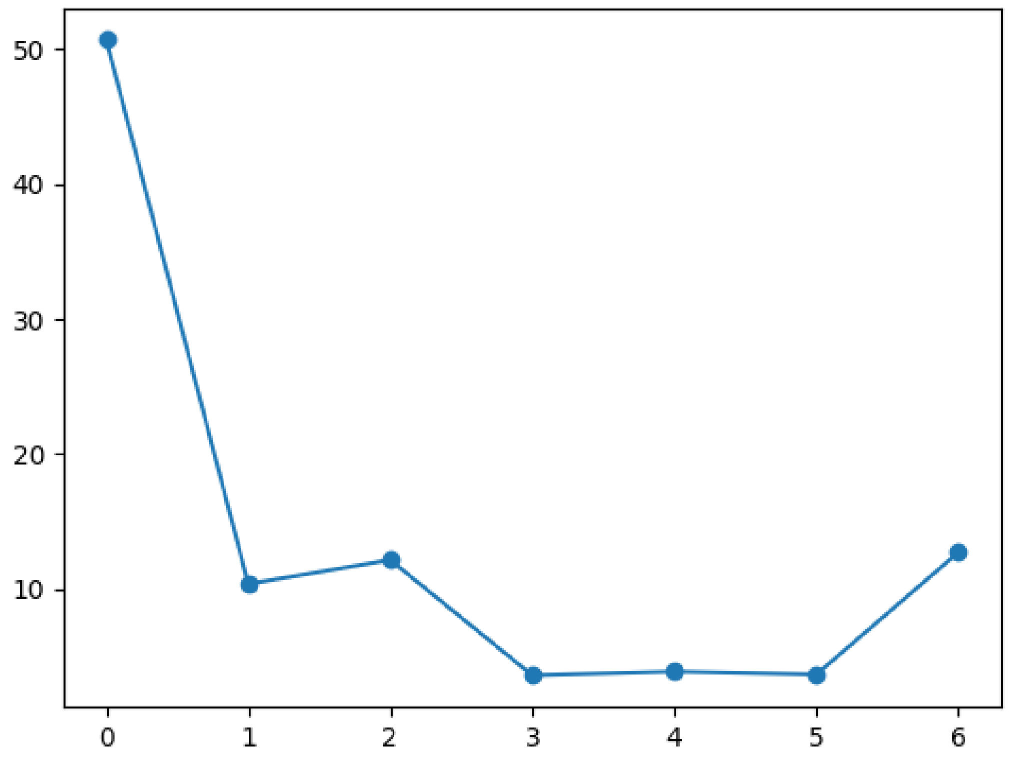
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2
−1
0
1
2



1.0
0.5
0.0
−0.5
−1.0
−1.5
−2
−1
0
1
2



1.0
0.5
0.0
−0.5
−1.0
−1.5
−2
−1
0
1
2



50
40
30
20
10
0
1
2
3
4
5
6

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression

training_data = np.loadtxt("train.csv", delimiter=",", dtype=np.float32)
testing_data = np.loadtxt("test.csv", delimiter= ",", dtype=np.float32)

x = training_data[:,0].reshape(-1,1)
y = training_data[:,1]

x_test = testing_data[:,0].reshape(-1,1)
y_test = testing_data[:,1]

errors_training = {}
errors_testing = {}

for degree in range(0,7):
    polynomial = PolynomialFeatures(degree=degree)
    x_poly = polynomial.fit_transform(x)

    reg = LinearRegression()
    reg.fit(x_poly, y)

    x_values = np.linspace(-2.2,2.2,100).reshape(-1,1)
    x_values_poly = polynomial.transform(x_values)

    y_values = reg.predict(x_values_poly)

    squared_error_training = np.sum((reg.predict(polynomial.transform(x)) -
y)**2)
    squared_error_testing = np.sum((reg.predict(polynomial.transform(x_test))
- y_test)**2)

    print(squared_error_testing)
    print(squared_error_training)

    errors_training[degree] = squared_error_training
    errors_testing[degree] = squared_error_testing

    plt.scatter(x, y)
    plt.plot(x_values, y_values, color = "r")
    plt.show()

    plt.scatter(x_test, y_test)
    plt.plot(x_values, y_values, color = "r")
    plt.show()
```

```python
x = errors_training.keys()
y = errors_training.values()

plt.scatter(x,y)
plt.plot(y)
plt.show()

x_test = errors_testing.keys()
y_test = errors_testing.values()

plt.scatter(x_test,y_test)
plt.plot(y_test)
plt.show()
```